WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, October 14, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L29	L28 and l22	6
☐	L28	(control$4 near5 robot) same application	1777
☐	L27	l1.ab. and l22	8
☐	L26	l2 and l22	0
☐	L25	l4 and l22	5
☐	L24	l5 and l22	0
☐	L23	l14 and L22	1
☐	L22	l18 or l19 or l20 or L21	4141
☐	L21	700/11,23.ccls.	2311
☐	L20	700/2,7.ccls.	416
☐	L19	713/100.ccls.	704
☐	L18	713/1.ccls.	1053
☐	L17	l14.ab.	8
☐	L16	l14.clm.	4
☐	L15	synchroniz$9 near2 start-up near2 control$4	14
☐	L14	synchroniz$9 near5 start-up near5 control$4	48
☐	L13	l7 and (application same synchroniz$9)	2
☐	L12	l7 and (control$4 same synchroniz$9)	1
☐	L11	l7 and (l1 same synchroniz$9)	0
☐	L10	l7 and synchroniz$9	15
☐	L9	l7 same synchroniz$9	0
☐	L8	l1 and L7	9
☐	L7	l4 same sequen$9	28
☐	L6	l1 same L4	0
☐	L5	l1 and L4	30
☐	L4	(initialization near2 step) same object	328
☐	L3	L2 same objects	8
☐	L2	((plurality or multiple or several) near3 initialization near2 steps)	85
☐	L1	(start$4 near2 application)	11528

END OF SEARCH HISTORY

L31: Entry 1 of 4 File: USPT Mar 28, 1995

DOCUMENT-IDENTIFIER: US 5402334 A
TITLE: Method and apparatus for pseudoperiodic drive

Detailed Description Text (63):
The present application is applicable to (I.) any driven system having an originally periodic drive, (II.) which is capable of multiple periods (period-doubling or tripling etc.), and/or (III.) which is capable of having multiple basins of attraction. Pseudoperiodic drives may be used with arrays of semiconductor detectors such as those used to detect optical, electrical, thermal or pressure signals, or with Josephson junction arrays used as voltage references or magnetic field detectors, when each element of such an array is driven by a common periodic signal such that period multiplication or multiple basins of attraction exist. A pseudoperiodic drive (consisting of an electrical, optical or magnetic signal) might be preferable to a periodic drive when it was necessary for all elements of such an array to be synchronized, i.e. each element must do the same thing at the same time. A pseudoperiodic drive might also be useful for synchronizing nonlinear chemical processing systems or chemical reactors which are driven by incoming streams of chemicals. Furthermore, the pseudoperiodic drive of the present invention is applicable to physiological or biosystems. For example, two similar valves in an artificial heart can be controlled with a pseudoperiodic signal. Furthermore, the pseudoperiodic signals can as well be implemented in pacemakers to control, in synchronization, the cells of an actual human heart. Also, pseudoperiodic drives can be used for artificial respirators. Additionally, the pseudoperiodic signals can be used for controlling artificial limbs or robots. For example, these pseudoperiodic signals can be used to control muscles in one's arm. The pseudoperiodic drive of the present invention is very applicable to biosystems because biosystems do not depend on complete accuracy or rely on an output occurring at a predictable instance of time. In other words, biosystems are more forgiving than sequential networks or traditionally known control systems and computer systems. Pseudoperiodic drives can also be used to provide a clock signal of a neural network. In conventional neural networks, the clock signal is periodic. Future neural networks may be implemented as "fuzzy", or inexact systems, where perfect periodic signals are not necessary, so that pseudoperiodic timing signals may be used to prevent different elements of the network from occupying different phases or attractors.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L29: Entry 3 of 6 File: USPT Dec 21, 1999

DOCUMENT-IDENTIFIER: US 6004019 A
TITLE: Integrated control system for a work robot

Detailed Description Text (13):
FIG. 2 illustrates a case where the robot control CPU 1 and the pressing force control CPU 2 are arranged in a single CPU 1', the PLC.cndot.CPU 4 and the network CPU 5 are arranged in a single CPU 4', and the welding electrical current control CPU 3 is a multi-task control CPU 3'. The multi-task (sealant coating, laser and vision) control CPU 3' is preferably extendable in its application. In the embodiment shown, a CCD camera 20 and its vision controller 21 are connected to the application-extended terminals.

Current US Cross Reference Classification (1):
700/2

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

End of Result Set

Generate Collection Print

L23: Entry 1 of 1 File: USPT Sep 14, 1999

DOCUMENT-IDENTIFIER: US 5951682 A
TITLE: Start-up system of a computer system

Detailed Description Text (94):
The start-up control programs use the start-up tables (as already mentioned) in order to control the start-up. Put more precisely, they control the synchronized start-up by transferring SP cells from one state into the other.

Current US Original Classification (1):
713/1

Previous Doc Next Doc Go to Doc#

L25: Entry 2 of 5 File: USPT Dec 31, 2002

DOCUMENT-IDENTIFIER: US 6502190 B1
TITLE: System and method for computer system initialization to maximize fault isolation using JTAG

Brief Summary Text (12):
It is therefore one object of the present invention to break down and set forth a sequence of steps for the initialization process or procedure of a complex computer system in order to accommodate detection of errors and failures and triggering of a respective attention or checkstop for and during the system initialization, set up, or power on.

Current US Cross Reference Classification (1):
713/1

L25: Entry 3 of 5 File: USPT Jan 9, 2001

DOCUMENT-IDENTIFIER: US 6173327 B1
TITLE: Object-oriented method and apparatus for information delivery

Detailed Description Text (103):
FIGS. 10a-10b shows the interaction between Adapter components. In step 1003, the initialization routines of the authenticator 161 and navigator 159 create objects 1610, 1590 to satisfy requests. The transporter 155 and the adapter 53 create Factory objects 1550, 1530. In step 1005, the dispatcher 151 calls the Factory object 1530 created by the adapter 153. (The dispatcher may call the Factory object many times depending upon the number of requests that the dispatcher is willing to serve. For this example, we assume that only one request can be handled.) In step 1007, the adapter, upon receiving the call to the Factory object, calls the transporter Factory object 1550. The transporter Factory object 1550, in step 1009, creates a transporter Stream object 1551. The transporter Factory object 1551 returns the transporter Stream object reference and handle to the adapter 153 in step 1011. The adapter 151, in step 1013, calls the transporter Stream object's Accept operation. The Accept operation, as discussed above, causes the transporter stream object 1551 to wait for incoming requests.

Detailed Description Text (110):
FIG. 12 describes the information provider implementation of Factory and Stream interfaces using a stream table and an object table stored in memory 40. In the start-up phase, at step 1203, the initialization routines of the implementation libraries are called. This routine creates an information provider Factory object 1390. If the trader is being used, the information provider will register with the trader by calling the trader object using the Register New Information Provider operation. In step 1205, the information provider receives a factory call from the dispatcher or from the trader. In the implementation of the information provider Factory, information in the call is used to select an information source and verify that the source can access the information. In step 1208, the Factory implementation selects a stream table entry for a new stream. The implementation generates a new unique handle for the stream and writes the stream context into the table entry. Next, in step 1209, the Factory implementation creates or selects an addressable Stream object 1391. The Stream object reference and handle are returned to the caller in step 1211.

Current US Cross Reference Classification (2):
713/100

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L27: Entry 3 of 8 File: USPT Apr 23, 2002

DOCUMENT-IDENTIFIER: US 6377977 B1
TITLE: Method for loading application program and opening files in host terminals before collaborating on a joint project

Abstract Text (1):
In a local area network, at least one host terminal has an application program and operates as a source host and other host terminals operate as destination hosts. Each host terminal executes a file management program. The source host, when executing its file management program, updates its management table with an application name identifying the application program and a file name, and transmits a request message to the destination hosts. Each destination host, when executing its file management program, responds to the request message by acquiring the application program from the source host if the application program is nonexistent in the destination host and updates its management table with the application name and the file name. Each host terminal starts the application program by opening a file identified by the file name and collaborates on a joint project with other host terminals.

Current US Cross Reference Classification (1):
713/1

Previous Doc Next Doc Go to Doc#

First Hit Previous Doc Next Doc Go to Doc#

End of Result Set

Generate Collection Print

L27: Entry 8 of 8 File: JPAB Jul 29, 1986

PUB-NO: JP361168004A
DOCUMENT-IDENTIFIER: JP 61168004 A
TITLE: PROGRAMMABLE CONTROLLER PROVIDED WITH APPLICATION INSTRUCTION

PUBN-DATE: July 29, 1986

INVENTOR-INFORMATION:

NAME	COUNTRY
HASHIMOTO, MASATO	
KATO, MASATO	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
MITSUBISHI ELECTRIC CORP	

APPL-NO: JP60007378
APPL-DATE: January 21, 1985

US-CL-CURRENT: 700/11
INT-CL (IPC): G05B 19/02; G06F 9/06

ABSTRACT:

PURPOSE: To improve the applicability from the viewpoint of constituting a ladder circuit by gathering information, which contacts in a circuit including an application instruction have, and combining and processing them in case of the application program.

CONSTITUTION: In case of the circuit including the application instruction, a programmable controller gathers and processes contact information and starts the operation of the application instruction. If an application instruction has a format to turn on a contact group 5 of contacts B0~Bn in accordance with the number of turned-on contacts of a contact group 1 of contacts A1~An, the controller searches turned-on contacts of the contact group and counts them. When all contacts are checked, the controller controls contacts on a basis of the check result so that some of contacts in the contact group 5 of contacts B0~Bn are turned on.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L17: Entry 2 of 8 File: USPT Nov 20, 1984

DOCUMENT-IDENTIFIER: US 4483236 A
TITLE: Pneumatic throttle control

Abstract Text (1):
A pneumatic throttle control configuration provides synchronized start-up and independent shut-off of multiple pneumatic tools. Each tool includes a throttle control assembly having a manually operable throttle valve which selectively admits air to the vane motor of the tool and a spring-biased piston operably coupled to the throttle valve which maintains it in an open position in response to air pressure in a control line which interconnects the tools. The vane motor includes a governor controlled valve which opens at a predetermined motor speed and provides air to the control line and the pistons of each tool of the multiple tool arrangement, thereby opening every throttle valve. Each pneumatic tool may then operate and stall independently. When the last tool stalls, the control air flow to all tools is terminated and the throttle valves close.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
Generate Collection Print

L17: Entry 3 of 8 File: USPT Nov 30, 1982

DOCUMENT-IDENTIFIER: US 4361826 A
TITLE: Clutch alarm system

Abstract Text (1):
Normally-open switches in series with a power source to an alarm device are closed by simultaneous actuation of a motor vehicle clutch and accelerator to provide an alarm to the operator. Means are provided for preventing the generation of an alarm during low-gear start-up and/or during synchronized operation of the controls for gear changing.

Previous Doc Next Doc Go to Doc#

L8: Entry 1 of 9 File: USPT Mar 23, 2004

DOCUMENT-IDENTIFIER: US 6710764 B1
TITLE: Method and system for processing force feedback effects generated at a host for playback at a physical interaction device

Detailed Description Text (69):
At this point, all objects and resources are set up properly in such a way that the application can start downloading/playing/modifying/stopping forces. The Simulation Thread is running and functional, the virtual device is initialized and empty. The next phase in the dynamic operation of the device driver software is the running phase. The running phase is described below and on FIG. 7.

Detailed Description Text (73):
As described above, the virtual device was created during the initialization (step 618) and it represents an ideal or perfect device which is presented to the application program and is infinitely capable of performing all requested feedback effects. Initially the virtual device object merely maintains its state (step 726) until a request for force feedback changes is received from the application program (step 728). There can be any sequence of the supported functions (Download (or Change) Effect, Destroy Effect, Start Effect, Stop Effect, Get Effect State, Send FF Command, Get Device Status, Set Overall Gain), which DirectInput receives from the application program and passes on to the driver. The force feedback driver, in turn, forwards those requests to the Virtual Device responsible for the device ID specified in the request (step 730) and allows it to address any device-specific parameters, and if required, returning the Virtual Device's error code. The Virtual Device is where the requests are being processed (step 732). None of these requests require any significant amount of processing time and return immediately after they show their effect. Once the virtual device has processed these requests, it forwards them to the simulation thread (step 734) for translation to low-level commands and subsequent communication to the physical device. Examples of some the single operations that may be performed at this level (step 732), and their corresponding results are shown below:

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L8: Entry 2 of 9 File: USPT May 27, 1997

DOCUMENT-IDENTIFIER: US 5634124 A
TITLE: Data integration by object management

Detailed Description Text (682):
With regard to the initiation of a data exchange operation, the first steps are the initialization of the operation, the identification of the location where the data is to appear in the destination object, and the identification of the data to be exchanged in the source object. As will be described, copy, move, and share operations are typically initiated manually by the system user, usually by a menu pick or by a command entered through the keyboard. The sequence of identification of the source object data and the destination object location for the data then depends upon whether the operation is a "from" or "to" operation, with the user's view of the operation moving between the source and destination objects are required by the sequence of the operation (i.e., the user can first identify the source data or the user can first identify the destination location).

Detailed Description Text (793):
This function will invoke the manager for the specified object and send the given request to it. The application will be given the link, any associated link specification, and caller-supplied request options. The caller can also ask to wait for a connection to be established with the invoked application for further communications. The function will return when the application has been started and has acknowledged the request.

Detailed Description Text (799):
Reply flag: TRUE if a reply is expected from the invoked application, FALSE if not. If TRUE, the function will wait until a reply has been received from the newly started application, successful or not.

Detailed Description Text (810):
This function initializes APPACK communications, so that the caller can process a startup request, perform matchmaker operations, or start other applications. It must be called before any other APPACK services are used. It will interpret the startup request sent to an application which has been invoked by the Application Manager, and return the contents to the caller. The function will return an operation ID to be used for further processing of this request. If the caller was not invoked by the Application Manager, the request code returned will be NONE, and no APOPID will be returned.

Detailed Description Text (891):
Reply flag: TRUE if a reply is expected from the invoked application, FALSE if not. If TRUE, the function will wait until a reply has been received from the newly started application, successful or not.

Previous Doc Next Doc Go to Doc#

L8: Entry 6 of 9 File: USPT Apr 12, 1994

DOCUMENT-IDENTIFIER: US 5303379 A
TITLE: Link mechanism for linking data between objects and for performing operations on the linked data in an object based system

Detailed Description Text (682):
With regard to the initiation of a data exchange operation, the first steps are the initialization of the operation, the identification of the location where the data is to appear in the destination object, and the identification of the data to be exchanged in the source object. As will be described, copy, move, and share operations are typically initiated manually by the system user, usually by a menu pick or by a command entered through the keyboard. The sequence cf identification of the source object data and the destination object location for the data then depends upon whether the operation is a "from" or "to" operation, with the user's view of the operation moving between the source and destination objects are required by the sequence of the operation (i.e., the user can first identify the source data or the user can first identify the destination location).

Detailed Description Text (792):
This function will invoke the manager for the specified object and send the given request to it. The application will be given the link, any associated link specification, and caller-supplied request options. The caller can also ask to wait for a connection to be established with the invoked application for further communications. The function will return when the application has been started and has acknowledged the request.

Detailed Description Text (798):
Reply flag: TRUE if a reply is expected from the invoked application, FALSE if not. If TRUE, the function will wait until a reply has been received from the newly started application, successful or not.

Detailed Description Text (809):
This function initializes APPACK communications, so that the caller can process a startup request, perform matchmaker operations, or start other applications. It must be called before any other APPACK services are used. It will interpret the startup request sent to an application which has been invoked by the Application Manager, and return the contents to the caller. The function will return an operation ID to be used for further processing of this request. If the caller was not invoked by the Application Manager, the request code returned will be NONE, and no APOPID will be returned.

Detailed Description Text (889):
Reply flag: TRUE if a reply is expected from the invoked application, FALSE if not. If TRUE, the function will wait until a replay has been received from the newly started application, successful or not.

L13: Entry 1 of 2 File: USPT Mar 23, 2004

DOCUMENT-IDENTIFIER: US 6710764 B1
TITLE: Method and system for processing force feedback effects generated at a host for playback at a physical interaction device

Detailed Description Text (49):
A third portion of the driver program is the CVDevice. This class implements the concept of a "Virtual Device". It is the model of what is presented to DirectInput as the connected device, hiding the details of the underlying hardware. CVDevice handles all the requests coming from DirectInput. Most of the operations are rather trivial at this level, with the only sophisticated one being DownloadEffect. It contains a handle to a mutex (mutual exclusion) object to synchronize interfering operations. It keeps all the relevant information about the device, most importantly a handle to allow ReadFile( ) and WriteFile( ) operations. It also keeps a collection of effects. Every time the application requests a download or an update of a downloaded effect, CVDevice makes a copy of all the information. Start and Stop operations cause a flag to be set for the applicable effect; SendForceFeedbackCommand sets the state in the CVDevice. Analogously, GetStatus and GetEffectState can immediately return the status of the device or the state of any given effect at any time. At any given time, CVDevice knows about any and all effects, their parameters and their state, and about the overall status of the device.

Detailed Description Text (73):
As described above, the virtual device was created during the initialization (step 618) and it represents an ideal or perfect device which is presented to the application program and is infinitely capable of performing all requested feedback effects. Initially the virtual device object merely maintains its state (step 726) until a request for force feedback changes is received from the application program (step 728). There can be any sequence of the supported functions (Download (or Change) Effect, Destroy Effect, Start Effect, Stop Effect, Get Effect State, Send FF Command, Get Device Status, Set Overall Gain), which DirectInput receives from the application program and passes on to the driver. The force feedback driver, in turn, forwards those requests to the Virtual Device responsible for the device ID specified in the request (step 730) and allows it to address any device-specific parameters, and if required, returning the Virtual Device's error code. The Virtual Device is where the requests are being processed (step 732). None of these requests require any significant amount of processing time and return immediately after they show their effect. Once the virtual device has processed these requests, it forwards them to the simulation thread (step 734) for translation to low-level commands and subsequent communication to the physical device. Examples of some the single operations that may be performed at this level (step 732), and their corresponding results are shown below:

L13: Entry 2 of 2 File: USPT Apr 11, 1989

DOCUMENT-IDENTIFIER: US 4821270 A
TITLE: Method for decoding data transmitted along a data channel and an apparatus for executing the method

Abstract Text (1):
Method of decoding data making it possible to correct certain errors, e.g. in synchronization and transmission. The method relates to decoding data transmitted in groups of blocks. Each block has n1+n2 bits, where n1 is the length of a corresponding information word. The information word is extended to an error protection block of n1+n2 bits by application of a linear error protection code and an off-set which indicates the position of the block within a group. Synchronization is established by incremental generation of a position-indicating syndrome from the block. If error correction is found to be necessary, it begins after synchronization. Synchronization is restarted if too many errors occur. A device for implementing the method includes a radio receiver (21), a microprocessor (22) for its control and a universal decoder (11) consisting on the one hand of a demodulation (13) and clock regeneration (12) component, the input of which is connected to the multiplex output of the radio receiver (21) and, on the other, of a broadcast data processing microprocessor (14), the output of which is connected to said microprocessor (22) for controlling the radio receiver.

Brief Summary Text (8):
The object of the invention is realized in that the decoding method comprises the steps of (a) in an initialization phase receiving a sequence of (n1+n2) data bits from said channel and therefrom generating a first syndrome by means of incremental operations, each incremental operation taking into account exactly one most recently received data bit; (b) comparing said first syndrome to respective standard syndromes, the latter corresponding to all possible off-set words; (c) in case of non-correspondence omitting the least recently received data bit from said (n1+n2) data bits and updating said first syndrome while accounting for a still more recently received data bit and disregarding a least recently received data bit until either in step (b) or in step (c) a correspondence occurs with a standard syndrome corresponding to one of said predetermined off-set words; (d) in a user phase following step (c) generating for each block of (n1+n2) bits received over said data channel an associated syndrome and comparing it with its position indicating standard syndrome(s), each next block having a presumed next following position in the sequence of groups; (e) in case of correspondence outputting a first strategy controlling signal, while outputting the block's information word to a user; in case of non-correspondence executing an error protection operation on the error-protected block of n1+n2 bits without its associated off-set word, while outputting a second strategy controlling signal; (f) and wherein a predetermined time-sequence of said first and second strategy controlling signals restarts step (a).

Brief Summary Text (11):
The principle of the decoding is that the beginning of the synchronization can only be effected if a certain number of blocks have been received in sequence error-free; however after synchronization has been established, a certain amount of

errors is acceptable in that within each block a certain error protection is effected. In the above-cited specification, a group contains four blocks, whereas each block contains an information word of 16 bits. The error protection is effected by means of a systematic code and an amount of redundancy of 10 bits. The redundancy used allows for correcting single bit errors and burst errors up to a maximum length of five bits. It is known by itself to use a particular amount of redundancy to implement various different error protection strategies. For example, one such strategy is to omit the full error correcting capability of a particular code, but to enhance the error detection capability. For example, a code with a bitwise minimum Hamming distance of four may be used either for single bit error correction, double bit error detection, or alternatively, for treble bit error detection. Even in a single application, certain code blocks may be treated differently from other code blocks. The advantage of a linear error protection code is that the sum of two code blocks again constitutes a code block. The code may be either systematic on the bit level or non-systematic on the bit level. In the former case a correct block has the information word contained within n1 specific bit positions. In the latter case, the information word must be retrieved from a larger number of bit positions (up to a maximum number of (n1+n2) bit positions).

Brief Summary Text (12):
After the initialization or synchronization phase has been successfully concluded, the user phase starts. In certain situations, the amount of error in the user phase increases in such a degree, that synchronization must be considered as lost. In that case, the system returns to the synchronization phase. The errors correctable and/or detectable could occur in particularly in a vehicle, e.g. a car fitted with a car radio, when it passes beneath a bridge. This protection is very effective in that at various predetermined positions in the bit stream (spaced at n1+n2 bits apart) the actual syndrome generated is compared to the appropriate standard syndrome(s). If the comparison yields equality, synchronization is correct and also transmission is considered errorfree. If the comparison does not yield equality, an error protection operation is effected. The result of such operation may be detection of one, two or more errors (considered as uncorrectable), correction of one or more bit errors, or a combination thereof. The type of error is stored in an error register means. The system continuously keeps book of those error types and makes a decision after processing of each block as to the expected quality of the transmission. In certain applications the strategy of this decision is adjustable. One particular strategy would be: each block with an error (either correctable or not) increases an error score. If a certain limit is attained, the system is considered out of synchronism. If a correct block is received, the error score is reset to zero. Other strategies are explained hereinafter.

Detailed Description Text (8):
Thus, in case of a reasonably low number of errors, at some time or another, the comparison in block 44 will give a positive result (1) and the system proceeds to block 46. Herein it is presumed that the previous window is correctly positioned; also, the identity of the previous standard syndrome is known. Therefore, the standard syndrome(s) for the next expected off-set word(s) is (are) chosen from the comparison register means. Note that this expectation may be unique (in the case of exactly N different off-set words). In other cases it could be ambiguous. For example, in the case of N=4 and seven off-set words, the number of possibilities for the expected off-set word syndrome could vary between one and four. Furthermore, in block 46 the next syndrome S(J+1) is calculated in exactly the same way as in block 42. After calculation, in block 50, the new syndrome S(J+1) is compared to the (one or more) selected syndromes from the comparison register means. In the same way as in block 44, the result can be positive or negative. If the result is negative, the system assumes that synchronization is absent and it returns to block 42. If the result is positive, the system assumes that synchronization is correct, and it may go to the data processing part proper or user phase under broken line 62. The degree of safety can be increased further in that the process of blocks 46 and 50 can be repeated one or more times, while a

secondary counter keeps track of the number of these repetitions. If the number reaches a predetermined level, the line 62 may be transgressed. In certain applications only a single correct comparison in block 44 would be sufficient, its output directly leading to block 48. On the lower side of broken line 62 there is assumed correctness of the two most recently received blocks (or of an even higher number in case the secondary counter cited hereabove had been used). The system can now transmit the data word(s) of one (the most recent) or more of these correct blocks to a user. This operation is illustrated by block 48 "BUS": the actual function of the bus subsystem is illustrated in FIG. 5. Next, the operations of blocks 46/50 are repeated in blocks 52/54. In block 54 the same comparison is made as in block 50, each time for the appropriate standard syndrome(s). If the comparison indicates no error, the system proceeds via block 68 to block 48, wherein the information word is made available to the user. The error strategy block 68 is considered later. In principle, the loop of blocks 48, 52, 54, 68 may be executed continually. In contradistinction to block 50, however, now also the error protection feature is taken into account. Thus, if there is a discrepancy (0 in block 54), the system goes to block 56. In block 56, an error protection operation on the error-protected block of (n1+n2) bits without its associated off-set word is effected while making use of the syndrome proper. The off-set word is ignored by subtracting it (which is the same as bit-wise EXCLUSIVE-ORING). In the particular application of the RDS-system, there are various possible signalizations

Previous Doc Next Doc Go to Doc#

L3: Entry 1 of 8 File: USPT Aug 31, 2004

DOCUMENT-IDENTIFIER: US 6783075 B2
TITLE: Portable hand-supportable data terminal with automatically-activated laser scanning bar code symbol reader integrated therein

Detailed Description Text (310):
Beginning at the START block of Main System Control Routine and proceeding to Block A of FIG. 23A1, the bar code symbol reading system is "initialized". This initialization step involves several steps, including: activating (i.e. enabling) system override detection circuit 301', first control circuit C.sub.1 (304'), oscillator circuit 301, the system override signal producing means 303, laser scanning circuit 308, and photoreceiving circuit 308; and deactivating (i.e. disabling) all subcircuits aboard ASIC chip 333 shown in FIGS. 22A1 through 22A4 that are not associated with the system override circuit 301, i.e. laser-based object detection circuit 307', A/D conversion circuitry 310, second control circuit C.sub.2, bar code presence detection circuit 311, third control module 314, symbol decoding module 319, data packet synthesis module 320, and data packet transmission circuit 321. During this initialization step, all timers T.sub.1, T.sub.2, T.sub.3, T.sub.4, and T.sub.5 are reset to t=0, the Decoded Symbol Data Buffer (e.g. maintained within the symbol decoding module 319) is initialized, and the "A.sub.3 =1 Flag" (monitored within the third control module C.sub.3) is cleared.